Logistic Regression

In [1]:
```
import pandas  as pd
import seaborn as sns
import warnings

warnings.filterwarnings('ignore')

% matplotlib inline
```

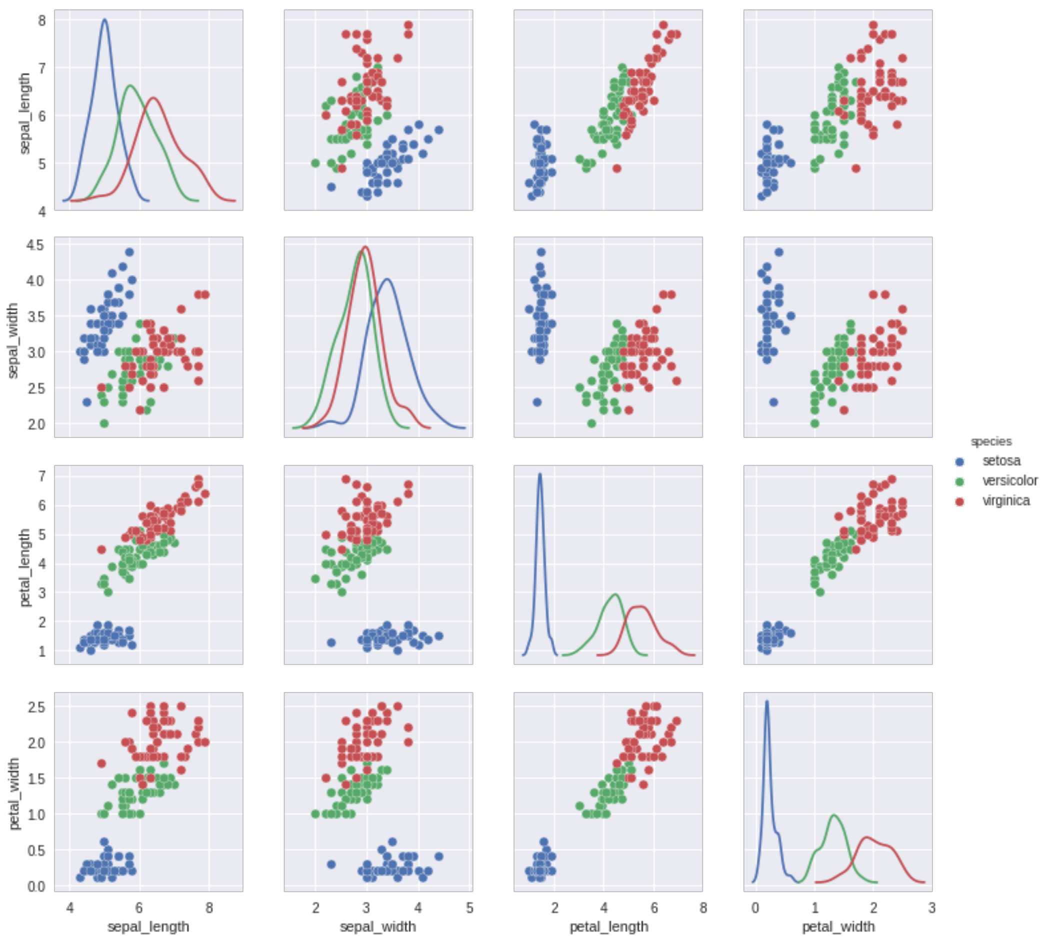
In [2]:
```
iris = sns.load_dataset('iris')
iris.sample()
```

Out[2]:

	sepal_length	sepal_width	petal_length	petal_width	species
105	7.6	3.0	6.6	2.1	virginica

In [3]:
```
sns.pairplot(iris, hue = 'species', diag_kind = 'kde')
```

Out[3]: <seaborn.axisgrid.PairGrid at 0x7f405116fac8>



In [4]:
```
from sklearn.preprocessing import LabelEncoder
```

In [5]:
```
columns = ['petal_length', 'species']
species = ['setosa', 'versicolor']

data    = iris[columns]
data    = data.loc[data['species'].isin(species)]
```

In [6]:
```
data.head(5)
```

Out[6]:

	petal_length	species
0	1.4	setosa
1	1.4	setosa
2	1.3	setosa
3	1.5	setosa
4	1.4	setosa

In [7]:
```
data.tail(5)
```

Out[7]:

	petal_length	species
95	4.2	versicolor
96	4.2	versicolor
97	4.3	versicolor
98	3.0	versicolor
99	4.1	versicolor

In [8]:
```
encoder         = LabelEncoder()
data['species'] = encoder.fit_transform(data['species'])
```

In [9]:
```
data.head(5)
```

Out[9]:

	petal_length	species
0	1.4	0
1	1.4	0
2	1.3	0
3	1.5	0
4	1.4	0

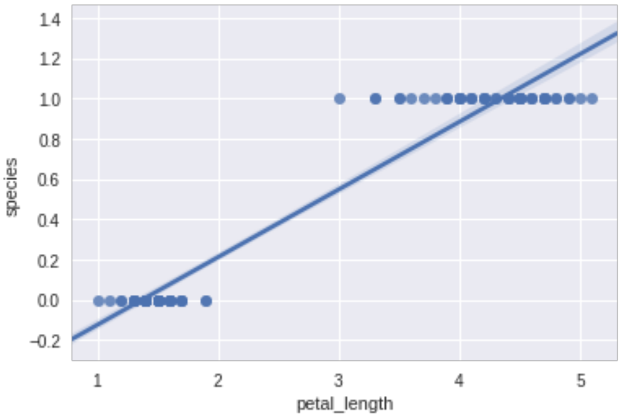
In [10]:
```
data.tail(5)
```

Out[10]:

	petal_length	species
95	4.2	1
96	4.2	1
97	4.3	1
98	3.0	1
99	4.1	1

In [11]:
```
sns.regplot(x = 'petal_length', y = 'species', data = data)
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3ffee63b38>

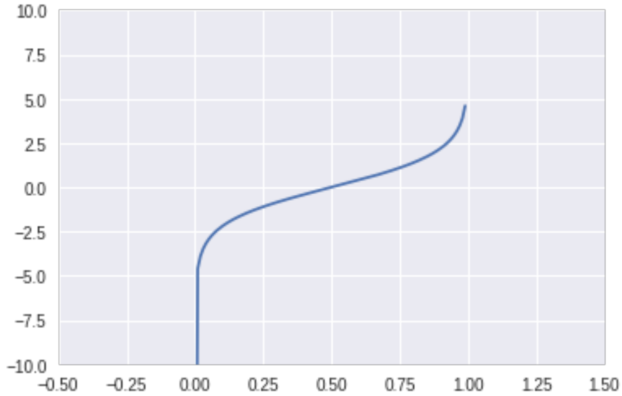


In [12]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [13]:
```
def logit(x):
    return np.log(x) - np.log(1 - x)
```

In [14]:
```
x  = np.arange(-1, 1, 0.01)

ax = plt.gca()
ax.set_xlim([-0.5, 1.5])
ax.set_ylim([ -10,  10])

plt.plot(x, logit(x))
```

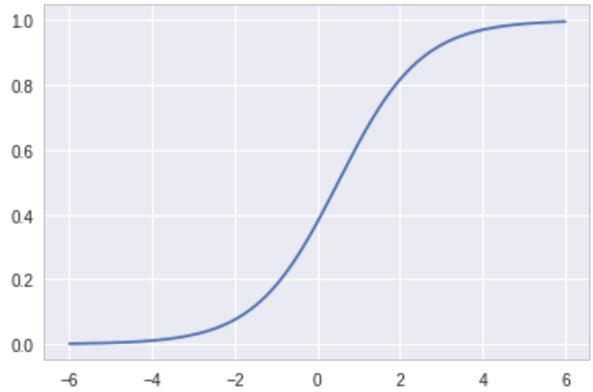
Out[14]: [<matplotlib.lines.Line2D at 0x7f3ffc851588>]



In [15]:
```
def logistic(x, max = 1, mid = 0.5, steepness = 1):
    return max / (1 + np.exp(-steepness * (x - mid)))
```

In [16]:
```
x = np.arange(-6, 6, 0.01)

plt.plot(x, logistic(x))
```

Out[16]: [<matplotlib.lines.Line2D at 0x7f3ffc7aae80>]



In [17]:
```
from sklearn.linear_model import LogisticRegression
```

In [18]:
```
X  = np.reshape(data['petal_length'], (len(data['petal_length']), 1))
y  = np.reshape(data['species'], (len(data['species']), 1))

lr = LogisticRegression()
lr.fit(X, y)
```

Out[18]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tr
ue,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001
,
          verbose=0, warm_start=False)
```

In [19]:
```
beta0 = lr.intercept_[0]
beta0
```

Out[19]: -4.1939275042884923

In [20]:
```
beta1 = lr.coef_[0,0]
beta1
```

Out[20]: 1.6673009365067675

In [21]:
```
def predict_prob(x):
    exponent = np.exp(beta0 + beta1 * x)

    return exponent / (1 + exponent)
```

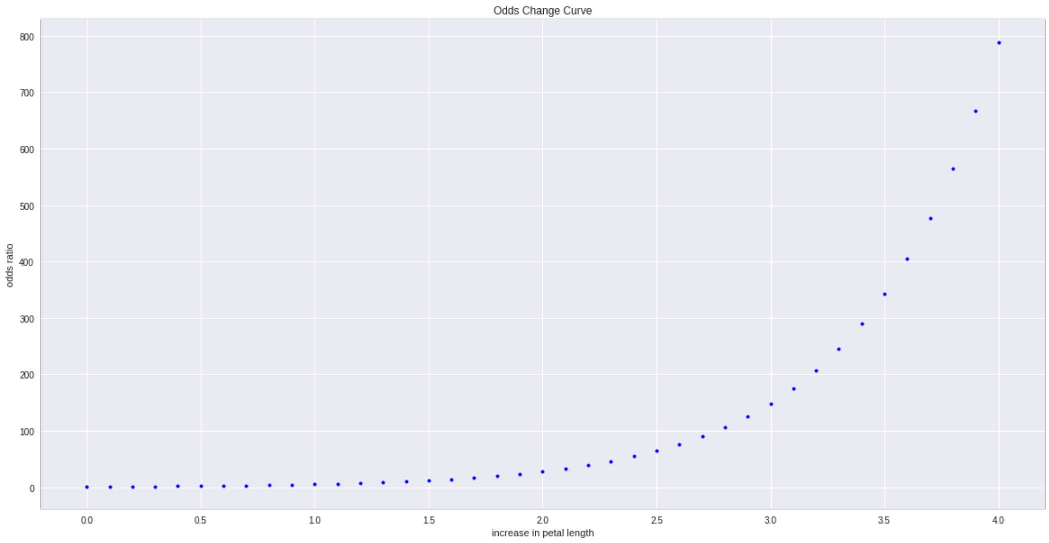
In [22]:
```
def odds(x):
    return x / (1 - x)
```

In [23]:
```
import prettytable as pt
from IPython.core.display import display, HTML
```

In [24]:
```
table  = pt.PrettyTable(['petal_length', '$\hat{p}$', '$1 - \hat{p}$', '
odds'])
sample = pd.concat([data['petal_length'].head(5), data['petal_length'].t
ail(5)])

for i in sample:
    p = np.round(predict_prob(i), 2)

    table.add_row([i, p, 1 - p, odds(p)])

display(HTML(table.get_html_string()))
```

petal_length	$\hat{p}$	$1 - \hat{p}$	**odds**
1.4	0.13	0.87	0.149425287356
1.4	0.13	0.87	0.149425287356
1.3	0.12	0.88	0.136363636364
1.5	0.16	0.84	0.190476190476
1.4	0.13	0.87	0.149425287356
4.2	0.94	0.06	15.6666666667
4.2	0.94	0.06	15.6666666667
4.3	0.95	0.05	19.0
3.0	0.69	0.31	2.22580645161
4.1	0.93	0.07	13.2857142857

In [25]:
```
incr  = 0.1
test  = np.arange(np.amin(data['petal_length']), np.amax(data['petal_length']), incr)
delta = 0
prob  = predict_prob(test[0])
oddsr = odds(prob) / odds(prob)

fig   = plt.figure(figsize = (20, 10))
plt.xlabel('increase in petal length')
plt.ylabel('odds ratio')

plt.title('Odds Change Curve')

plt.scatter(delta, oddsr, color = 'b', marker = '.')

for i in range(1, len(test)):
    q      = predict_prob(test[i])

    delta += incr
    oddsr  = odds(q) / odds(prob)

    plt.scatter(delta, oddsr, color = 'b', marker = '.')
```



In [26]:
```
predict            = data
predict['species'] = data['species'].apply(predict_prob)
```

In [27]:
```
print('petal length being either setosa or versicolor: ', -beta0/beta1)
```

```
petal length being either setosa or versicolor:  2.51539923745
```